



The RDKit and PostgreSQL: an open-source database system for chemistry

Gregory Landrum, Andy Palmer

NIBR IT

Novartis Institutes for BioMedical Research, Basel and Cambridge

5th Meeting on U.S. Government Chemical Databases and Open Chemistry
August 2011



Acknowledgements

- Novartis:
 - Tom Digby (Legal)
 - John Davies (CPC/LFP)
 - Eddie Cao (NIBR IT)
 - Richard Lewis (GDC/CADD)
 - Nikolaus Stiefl (GDC/CADD)
 - Peter Gedeck (GDC/CADD)
- Rational Discovery:
 - Santosh Putta (currently at Nodality)
 - Julie Penzotti
- RDKit open-source community
- postgresql cartridge:
 - Michael Stonebraker
 - Oleg Bartunov
 - Teodor Sigaev
 - Pavel Velikhov
- Entagen (SWIG wrappers):
 - Chris Bouton
 - Erik Bakke
 - James Hardwick
- knime.com
 - Michael Berthold
 - Thorsten Meinl
 - Bernd Wiswedel

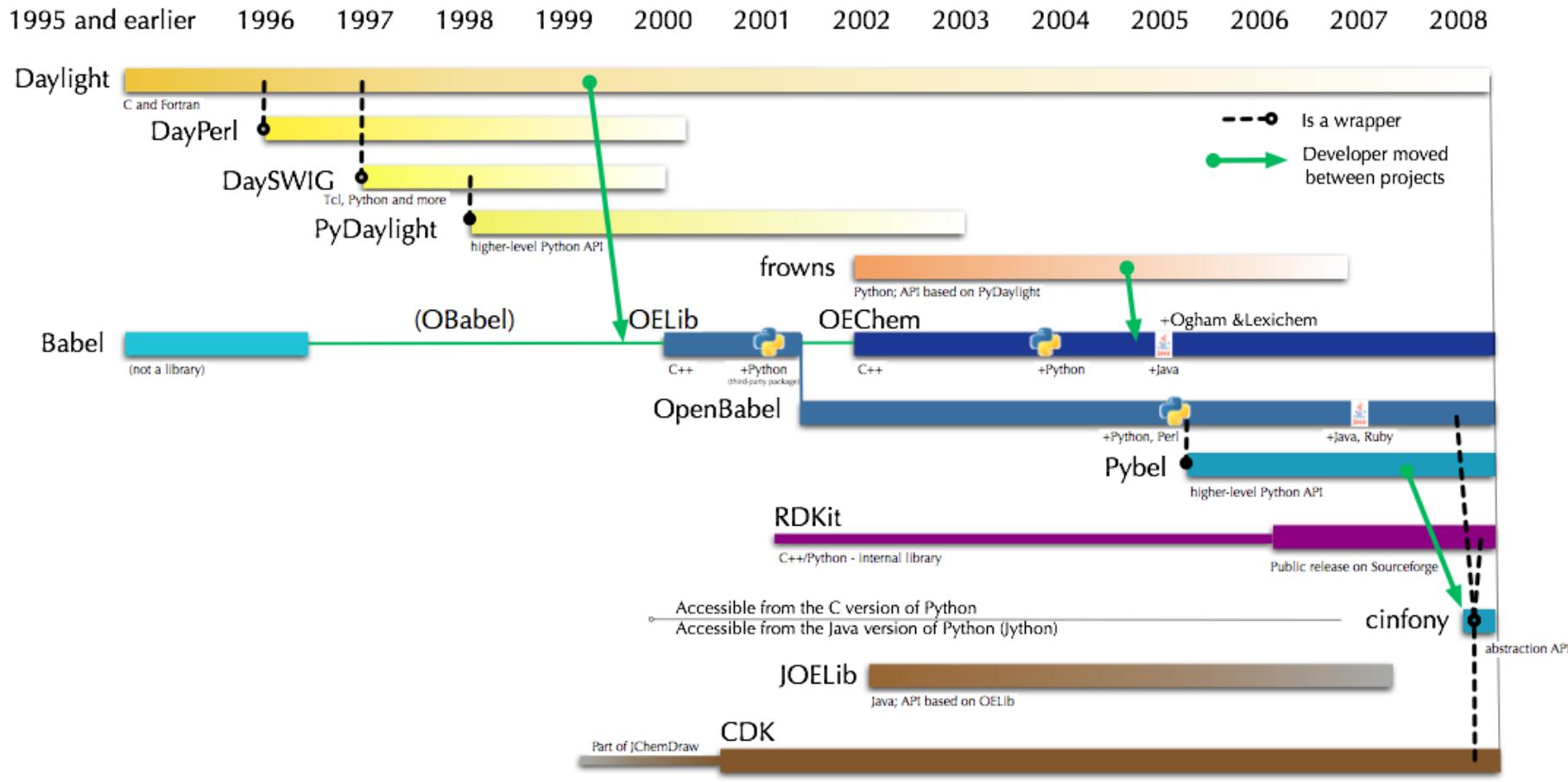
Overview

- **RDKit: what is it?**
- RDKit + Knime
- RDKit + PostgreSQL
- Case study: matched pairs analysis

Cheminformatics toolkits

Timeline of cheminformatics toolkits*

*(runs on Unix and supports SMILES and SMARTS)



Source: Andrew Dalke

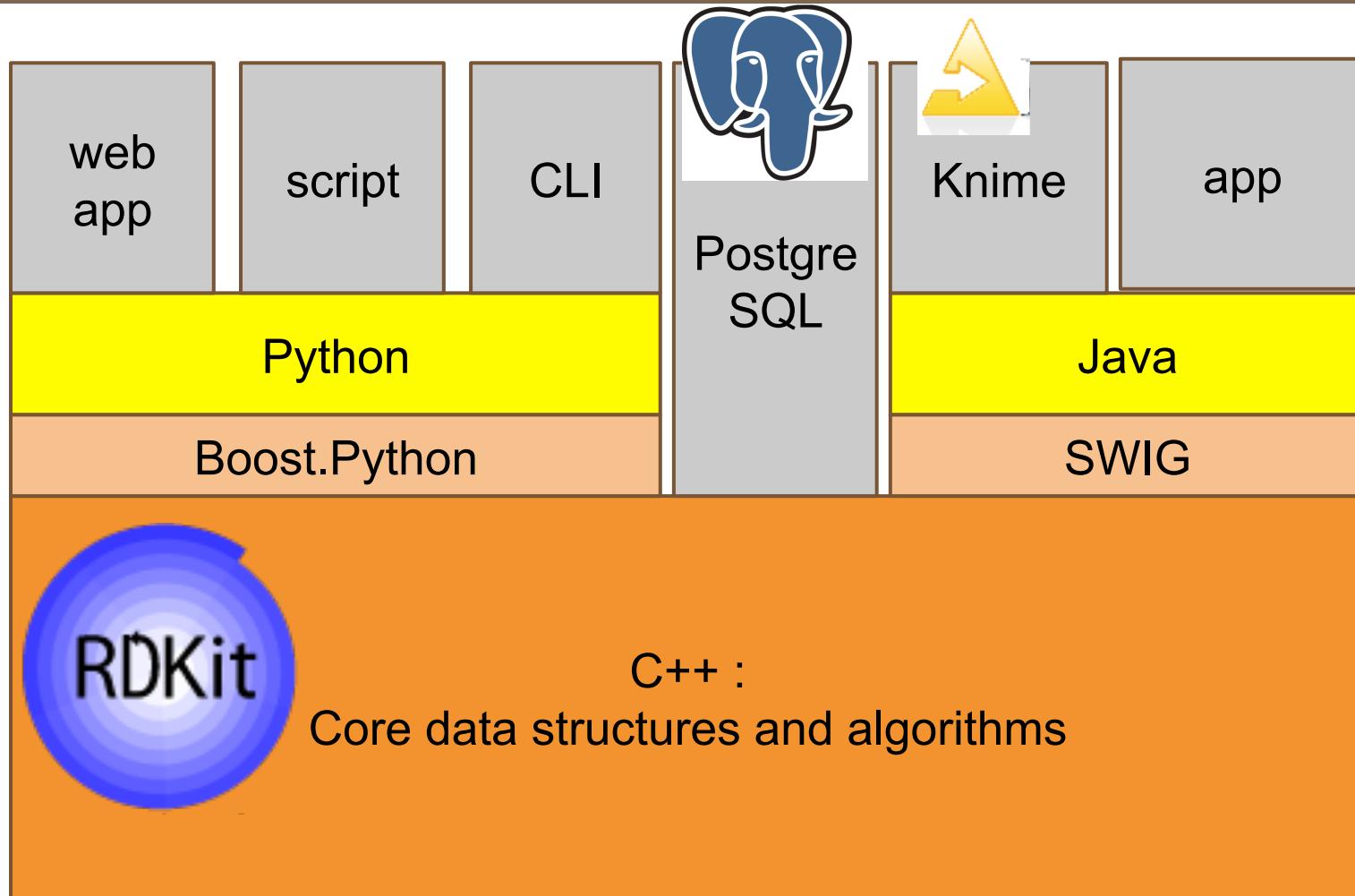
<http://www.dalkescientific.com/writings/diary/archive/2008/09/20/euroqsar.html>



RDKit: What is it?

- Python (2.x), Java, and C++ toolkit for cheminformatics
 - Core data structures and algorithms in C++, including heavy use of Boost
 - Python wrapper generated using Boost.Python
 - Java wrapper generated with SWIG
- Functionality:
 - 2D and 3D molecular operations
 - Descriptor generation for machine learning
 - Molecular database cartridge
 - Knime nodes
 - Supports Mac/Windows/Linux
- History:
 - 2000-2006: Developed and used at Rational Discovery for building predictive models for ADME, Tox, biological activity
 - June 2006: Open-source (BSD license) release of software, Rational Discovery shuts down
 - to present: Open-source development continues, use within Novartis, contributions from Novartis back to open-source version

What is this all about?



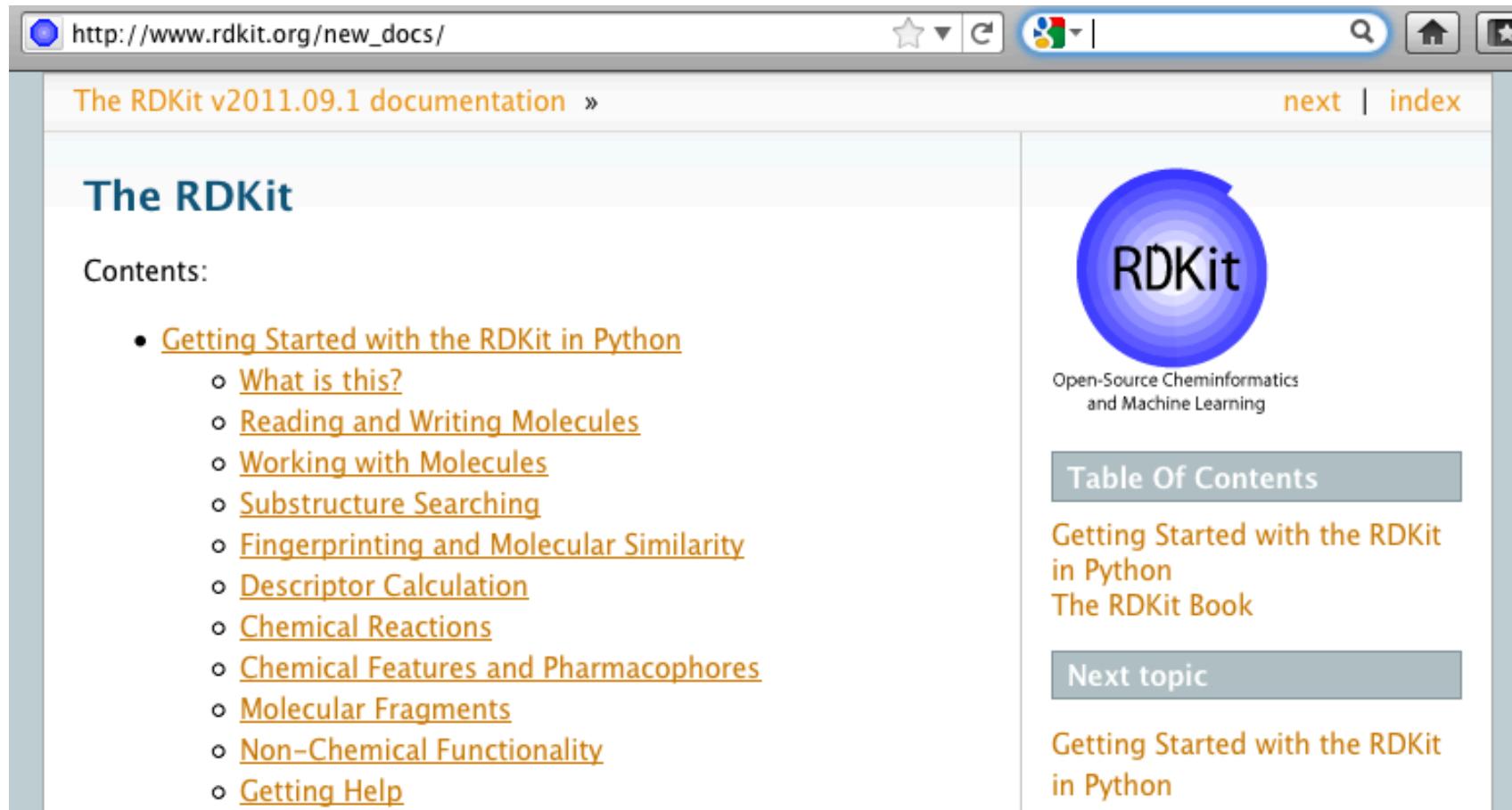
Exact same algorithms/implementations accessible from many different endpoints

RDKit: Where is it?

- Web page: <http://www.rdkit.org>
- Sourceforge: svn repository, bug tracker, mailing lists, downloads
 - <http://sourceforge.net/projects/rdkit>
- Google code: wiki, downloads
 - <http://code.google.com/p/rdkit/>
- Releases: quarterly (more or less)
- Licensing: new BSD
- Documentation:
 - HTML/PDF “Getting Started” documentation
 - in-code docs extracted by either doxygen (C++) or epydoc (python)
- Getting help:
 - Check the wiki and “Getting Started” document
 - The rdkit-discuss mailing list

RDKit: Documentation?

The new documentation:



A screenshot of a web browser displaying the RDKit v2011.09.1 documentation. The URL in the address bar is http://www.rdkit.org/new_docs/. The page title is "The RDKit". On the left, there's a "Contents" sidebar with a list of topics under "Getting Started with the RDKit in Python". On the right, there's a large circular logo for RDKit with the text "RDKit" in white, followed by "Open-Source Cheminformatics and Machine Learning". Below the logo are buttons for "Table Of Contents", "Getting Started with the RDKit in Python", "The RDKit Book", and "Next topic".

http://www.rdkit.org/new_docs/

The RDKit v2011.09.1 documentation »

next | index

The RDKit

Contents:

- [Getting Started with the RDKit in Python](#)
 - [What is this?](#)
 - [Reading and Writing Molecules](#)
 - [Working with Molecules](#)
 - [Substructure Searching](#)
 - [Fingerprinting and Molecular Similarity](#)
 - [Descriptor Calculation](#)
 - [Chemical Reactions](#)
 - [Chemical Features and Pharmacophores](#)
 - [Molecular Fragments](#)
 - [Non-Chemical Functionality](#)
 - [Getting Help](#)

Built using Python's standard docs tool: Sphinx

RDKit: Documentation?

Sample section from introductory docs:

Reading and Writing Molecules

Reading single molecules

The majority of the basic molecular functionality is found in module [rdkit.Chem](#):

```
>>> from rdkit import Chem
```

Individual molecules can be constructed using a variety of approaches:

```
>>> m = Chem.MolFromSmiles('Cc1ccccc1')
>>> m = Chem.MolFromMolFile('data/input.mol')
>>> stringWithMolData=file('data/input.mol','r').read()
>>> m = Chem.MolFromMolBlock(stringWithMolData)
```

All of these functions return a [Mol](#) object on success:

```
>>> m
<rdkit.Chem.rdcchem.Mol object at 0x...>
```

Molecules

- [Working with Molecules](#)
- [Substructure Searching](#)
- [Fingerprinting and Molecular Similarity](#)
- [Descriptor Calculation](#)
- [Chemical Reactions](#)
- [Chemical Features and Pharmacophores](#)
- [Molecular Fragments](#)
- [Non-Chemical Functionality](#)
- [Getting Help](#)
- [Advanced Topics/Warnings](#)
- [Miscellaneous Tips and Hints](#)
- [List of Available Descriptors](#)
- [List of Available Fingerprints](#)
- [Feature Definitions Used in the Morgan Fingerprints](#)
- [License](#)

The RDKit Book

[Previous topic](#)

Note: docs that include python code snippets are *tested*.

RDKit: Documentation?

The wiki: <http://code.google.com/p/rdkit/w/list>

PageName ▾	Summary + Labels ▾	Changed ▾	ChangedBy ▾
WorkingBuilds	Summary of platforms where the RDKit has been built. Development	23 hours ago	greg.landrum
GettingStartedInCPP_1	A sample application in C++ Tutorial CPlusPlus	Aug 14	greg.landrum
LoadingChEMBL	Loading the full ChEMBL database into PostgreSQL cartridge Notes	Aug 12	greg.landrum
BuildingModelsUsingDescriptors1	Building simple predictive models using descriptors Tutorial ML	Aug 10	greg.landrum
TrainAThreeClassSolubilityModel	Train a DecisionTree based on the Huuskonen data set	Jul 29	paul@tonair.de
DatabaseCreation2	Creating and using a chemical database 2: Working with the eMolecules catalog. cartridge Tutorial InProgress	Jul 5	greg.landrum
UsingTheCartridgeFromPython	Using the database cartridge from python Tutorial cartridge InProgress	Jul 4	greg.landrum
BuildingOnWindows	Building on Windows Development Notes	Jul 1	greg.landrum
BuildingWithCmake	New Linux (and Mac) build instructions Development Notes	Jun 30	greg.landrum
BuildingOnCentOS	RDKit build walk through for CentOS 5.5 Notes	Jun 13	greg.landrum
Benchmarking	Some benchmarking results Notes Development	Jun 10	greg.landrum
InstallingOnWindows	Getting a binary installation working on windows Tutorial	Jun 10	greg.landrum
GettingInvolved	Getting involved in the project.	May 15	greg.landrum
BuildingTheCartridge	How to build the PostgreSQL cartridge cartridge	May 14	greg.landrum
BuildingModelsUsingFingerprints1	Building simple predictive models using fingerprints Tutorial ML	May 2	greg.landrum
BenchmarkingMoleculeConstruction	Aspects of building a molecule Development Notes	Apr 5	greg.landrum
ExampleStructureQueries	Some example queries for doing structure searching. cartridge Tutorial	Apr 5	giallu
DescriptorsInTheRDKit	Overview of the descriptors available in the RDKit	Mar 25	greg.landrum

RDKit: Who is using it?

- Hard to say with any certainty
- ~300 downloads of each new version
- Active contributors to the mailing list from:
 - Big pharma
 - Small pharma/biotech
 - Software/Services
 - Academia
- Starting to see contributions coming from the community (wiki pages, code patches, changes to the build system, etc.) as well as active use in other systems.

Things you should know

- Molecules should be “correct”: i.e. there should be a valid Lewis-dot structure. If not, they will be rejected:

```
>>> Chem.MolFromSmiles('CC(F)(Cl)(Br)I')
[08:58:09] Explicit valence for atom # 1 C, 5, is greater
than permitted
```

- The software generally doesn’t try and read the user’s mind

Nitro groups and N-oxides are repaired:

```
>>> Chem.CanonSmiles('CN(=O)=O')
'C[N+](=O)[O-]'
>>> Chem.CanonSmiles('c1ccccc1=O')
'[O-][n+]1ccccc1'
```

but some odd constructs (this one from ChEMBL SMILES) are not:

```
>>> Chem.MolFromSmiles('CN=N#N')
[16:30:08] Explicit valence for atom # 2 N, 5, is greater
than permitted
... snip ...
```

```
>>> Chem.CanonSmiles('CN=[N+]=[N-]')
'CN=[N+]=[N-]'
```

What can you do with it?

A laundry list

- Input/Output: SMILES/SMARTS, SDF, TDT, SLN¹, Corina mol2¹, InChI²
- “Cheminformatics”:
 - Substructure searching
 - Canonical SMILES
 - Chirality support (i.e. R/S or E/Z labeling)
 - Chemical transformations (e.g. remove matching substructures)
 - Chemical reactions
 - Molecular serialization (e.g. mol <-> text)
- 2D depiction, including constrained depiction
- 2D->3D conversion/conformational analysis via distance geometry
- UFF implementation for cleaning up structures
- Fingerprinting:
Daylight-like, atom pairs, topological torsions, Morgan algorithm, “MACCS keys”, etc.
- Similarity/diversity picking
- 2D pharmacophores¹
- Gasteiger-Marsili charges
- Murcko analysis
- Hierarchical subgraph/fragment analysis
- RECAP and BRICS implementations

¹ functional, but not great implementations

² Solid output, experimental input

What can you do with it?

A laundry list, ctd

- Feature maps
- Shape-based similarity
- Molecule-molecule alignment
- Shape-based alignment (subshape alignment)¹
- Integration with PyMOL for 3D visualization
- Database integration
- Molecular descriptor library:
 - Topological (κ_3 , Balaban J, etc.)
 - Electrotopological state (Estate)
 - clogP, MR (Wildman and Crippen approach)
 - “MOE like” VSA descriptors
 - Feature-map vectors
- Machine Learning:
 - Clustering (hierarchical)
 - Information theory (Shannon entropy, information gain, etc.)
 - Decision trees, *naïve Bayes*¹, *kNN*¹
 - Bagging, random forests
 - Infrastructure (data splitting, shuffling, enrichment plots, serializable models, etc.)

¹ functional, but not great implementations

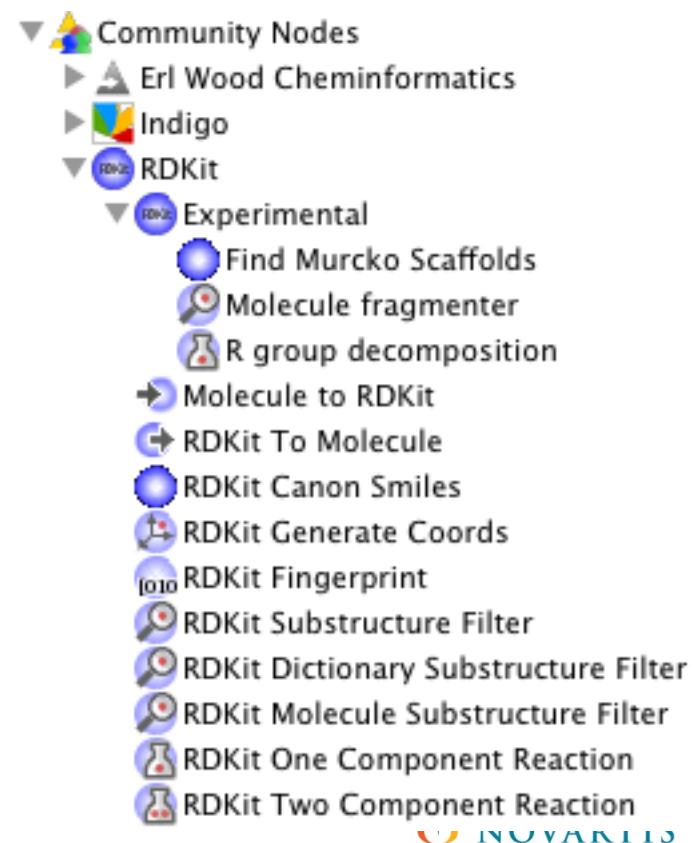
Overview

- RDKit: what is it?
- **RDKit + Knime**
- RDKit + PostgreSQL
- Case study: matched pairs analysis

Knime integration¹

- Out of the box Knime is strong on data processing and mining, weak on chemistry.
- Goal: develop a set of *open-source* RDKit-based nodes for Knime that provide basic cheminformatics functionality
- Distributed from knime community site
- Binaries available as an update site (no RDKit build/installation required)
- Work in progress: more nodes being added

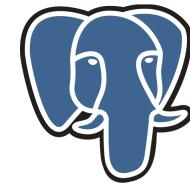
Note: We'll be developing more open-source nodes at the Knime open-source days in October:
<http://www.knime.org/kos-2011>



¹ Work done together with knime.com

Overview

- RDKit: what is it?
- RDKit + Knime
- **RDKit + PostgreSQL**
- Case study: matched pairs analysis



PostgreSQL

+



The database cartridge

- Integration of RDKit fingerprinting and substructure search functionality with PostgreSQL
- Bit-vector and counts-based fingerprints along with common similarity metrics (Tanimoto and Dice) integrated with PostgreSQL indexing system to allow fast searches (~1 million compounds/sec on a single CPU)
- Available similarity fingerprints:
 - Morgan (ECFP-like), available as bit vectors or counts
 - FeatMorgan (FCFP-like), available as bit vectors or counts
 - RDKit (Daylight-like), available as bit vectors
 - atom pairs, available as bit vectors or counts
 - topological torsions, available as bit vectors or counts
- SMILES- and SMARTS-based substructure querying integrated with indexing system
- Standard Lipinski-like descriptors (logP, tPSA, MW, etc.)
- Part of the RDKit open-source distribution since July 2010

Cartridge performance #1

- Database: 100K diverse drug-like molecules from ZINC
 - Molecules load/index time: 109 sec / 343 sec
 - Fingerprints (Morgan2) calculation/index time: 23.1 sec / 9.3 sec
- "Fragments" queries: 500 diverse fragment-like molecules from ZINC
- "Leads" queries: 500 diverse lead-like molecules from ZINC
- Hardware: MacBook Pro (2.5GHz Core2 Duo)
- Do queries via a cross join (i.e. 500 queries x 100K database molecules = 50M possible comparisons/searches)
- Results:

Query Set	SSS	Run time (sec)		
		Similarity (0.8)	Similarity (0.6)	Similarity (0.5)
Zinc Fragments	23.3	14.6	36.4	37.1
Zinc Leads	8.2	14.8	36.2	38.5

Cartridge performance #2

- Database: 5 million molecules from the emolecules catalog
 - Molecules load/index time: 4040 sec / 7832 sec
- Hardware: Dell Studio XPS (2.9GHz i7 with 8GB of RAM, Ubuntu Linux)

Similarity queries:

zinc lead-like set: queries return 0-50 hits in 5-8 seconds

SSS Query	Count	Run time (sec)
c1cccc2c1nncc2	996	2.4
c1ccnc2c1nccn2	1366	0.8
c1cncc2n1ccn2	523	2.3
Nc1ncnc(N)n1	14227	3.8
c1scnn1	88564	31.6
c1cccc2c1ncs2	95288	55.3

- Performance here is a function of the SSS fingerprint quality
- SMARTS-based queries can be considerably slower
- Getting first 50 results from any of these queries: <0.5 sec

The database cartridge

- How hard is it to add something new?
 1. Write C++ function implementing new functionality (probably not cartridge specific, could also be used from Python or Java)
 2. Add one boilerplate C++ function to cartridge code exposing that function
 3. Add one boilerplate C function to cartridge code
 4. Add one boilerplate SQL function to cartridge code
 5. Add tests
- The first step is the hardest part (unless the function already exists), the rest takes 5-10 minutes.
- Reference/HowTo for this:
<http://www.mail-archive.com/rdkit-discuss@lists.sourceforge.net/msg01549.html>

Using the database cartridge

Similarity search with Morgan fingerprint:

```
vendors=# select \
  id,tanimoto_sml(morganbv_fp('N=C1OC2=C(C=CC=C2)C=C1',2),mfp2) \
  from fps where morganbv_fp('N=C1OC2=C(C=CC=C2)C=C1',2)%mfp2 ;
  id      |    tanimoto_sml
-----+-----
 9171448 |  0.538461538461538
 765434  |  0.538461538461538
(2 rows)
```

Substructure Search:

```
vendors=# select count(*) from mols where mol@>'N=C1OC2=C(C=CC=C2)C=C1';
  count
-----
 2854
(1 row)
```

Properties:

```
vendors=# select mol_tpsa('C1CCC1CC(=O)O'::mol) tpsa, mol_logp('C1CCC1CC(=O)
O'::mol) clogp;
  tpsa |  clogp
-----+-----
 37.3 |  1.2612
(1 row)
```

Overview

- RDKit: what is it?
- RDKit + Knime
- RDKit + PostgreSQL
- **Case study: matched pairs analysis**

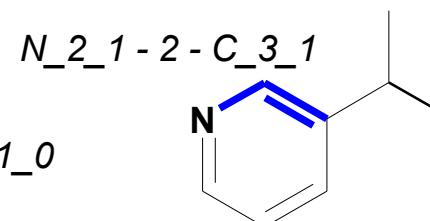
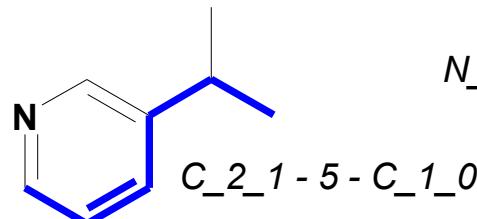
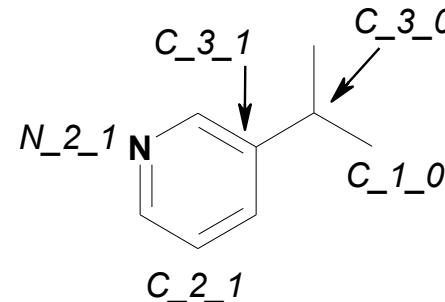
Case Study: bringing the pieces together

- Combine the RDKit + the PostgreSQL cartridge + Knime to do matched-pairs analysis
- Idea: find pairs of molecules that are structurally similar but that have quite different activities to identify interesting/useful transformations.
- Key concept is disparity: $\Delta\text{Activity} / (1\text{-similarity})$
- Easily done from Python using the RDKit, but it becomes time consuming as the number of molecules increases (N^2 similarity calculations required).

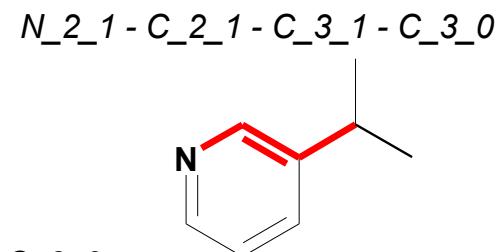
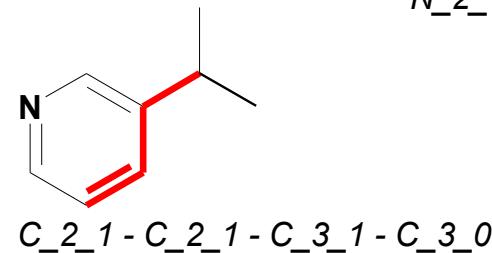
Atom-Pair and Topological-Torsion Fingerprints

related descriptors from the “distant” past

- Atom-type:
(Element, #heavy neighbors, #pi electrons)
- Atom Pair¹:
Atom-type – topological distance – Atom-type



- Topological Torsion²:
Atom-type – Atom-type – Atom-type – Atom-type

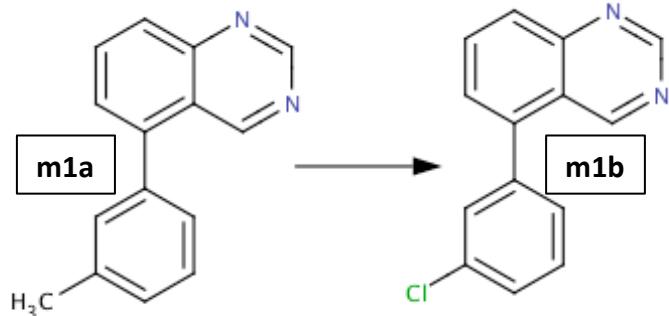


- Both fingerprints can use counts (not 1/0 values)

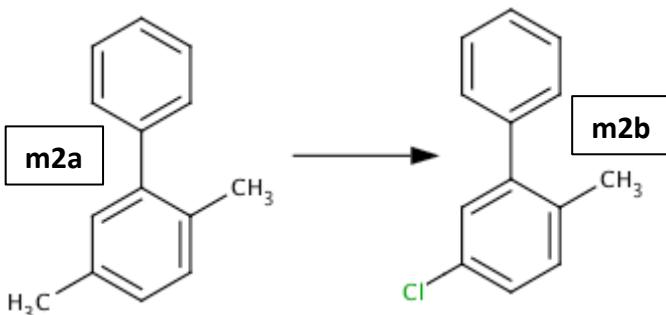
1 R.E. Carhart, D.H. Smith, R. Venkataraghavan *JC/CS* **25**:64-73 (1985)

2 R. Nilakantan, N. Bauman, J. S. Dixon, R. Venkataraghavan; *JC/CS* **27**:82-5 (1987).

Using count-based fingerprints for matched pairs

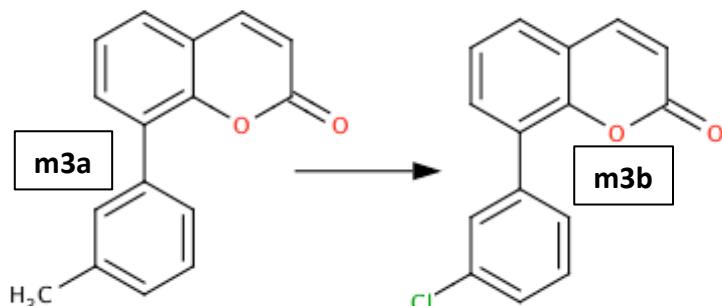


```
>>> DataStructs.TanimotoSimilarity(fp1a, fp1b)  
0.89189189189189189  
>>> DataStructs.TanimotoSimilarity(fp2a, fp2b)  
0.85185185185185186  
>>> DataStructs.TanimotoSimilarity(fp3a, fp3b)  
0.89743589743589747
```



```
>>> f1 = fp1a-fp1b  
>>> f2 = fp2a-fp2b  
>>> f3 = fp3a-fp3b  
>>> f1==f2  
True  
>>> f1==f3  
True
```

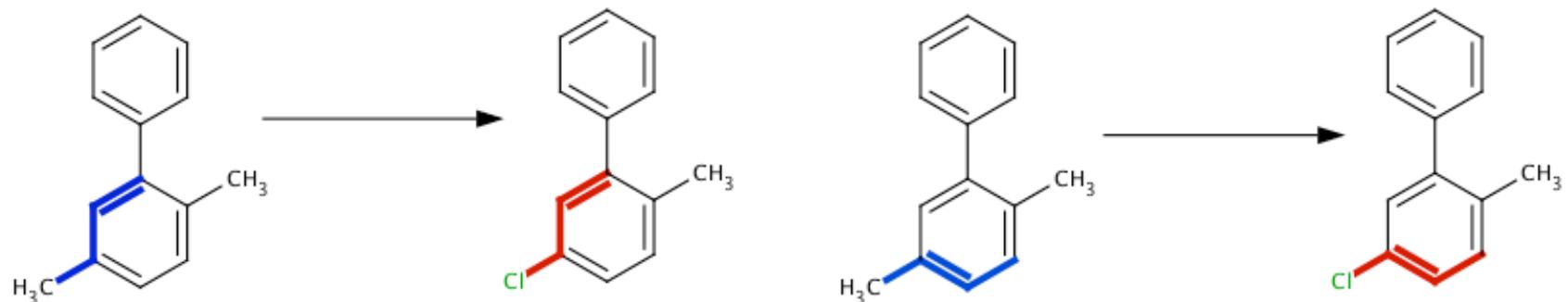
Fingerprint math:
subtracting two
fingerprints from
another gives another
fingerprint



```
>>> f1.GetNonzeroElements()  
5513433632L: 1, 5647651360L: 1, 34370506793L: -1,  
34370506794L: -1}  
>>> for bit,v in f1.GetNonzeroElements().items(): print  
Torsions.ExplainPathScore(bit),v  
.....:  
((('C', 1, 0), ('C', 3, 1), ('C', 2, 1), ('C', 2, 1)) 1  
((('C', 1, 0), ('C', 3, 1), ('C', 2, 1), ('C', 3, 1)) 1  
((('C', 3, 1), ('C', 2, 1), ('C', 3, 1), ('Cl', 1, 0)) -1  
((('C', 2, 1), ('C', 2, 1), ('C', 3, 1), ('Cl', 1, 0)) -1
```

Using count-based fingerprints for matched pairs

```
>>> for bit,v in f1.GetNonzeroElements().items():
.....:     print Torsions.ExplainPathScore(bit),v
.....:
((C', 1, 0), ('C', 3, 1), ('C', 2, 1), ('C', 2, 1)) 1
((C', 1, 0), ('C', 3, 1), ('C', 2, 1), ('C', 3, 1)) 1
((C', 3, 1), ('C', 2, 1), ('C', 3, 1), ('Cl', 1, 0)) -1
((C', 2, 1), ('C', 2, 1), ('C', 3, 1), ('Cl', 1, 0)) -1
```



A standard molecular db schema

Table "herg_data.mols"

Column	Type	Modifiers
id	integer	not null
compound_id	text	
m	mol	

Indexes:

"mols_pkey" PRIMARY KEY, btree (id)
"molidx" gist (m)

Triggers:

process_mol AFTER INSERT OR DELETE OR UPDATE ON
herg_data.mols FOR EACH ROW EXECUTE PROCEDURE
herg_data.process_update_mol()

Table "herg_data.molvals"

Column	Type	Modifiers
id	integer	not null
pIC50	float	
ACTIVITY_CLASS	text	
CompoundName	text	
MDLPublicKeys	text	

Indexes:

"molvals_pkey" PRIMARY KEY, btree (id)

Table "herg_data.countfps"

Column	Type	Modifiers
id	integer	not null
pairfp	sfp	
torsionfp	sfp	
mfp2	sfp	

Indexes:

"countfps_pkey" PRIMARY KEY, btree (id)
"apfpcountidx" gist (pairfp sfp_low_ops)
"mfpscountidx" gist (mfp2 sfp_low_ops)
"torsionfpcountidx" gist (torsionfp sfp_low_ops)

Table "herg_data.fps"

Column	Type	Modifiers
id	integer	not null
mfp2	bfp	

Indexes:

"fps_pkey" PRIMARY KEY, btree (id)
"ffp2idx" gist (ffp2)
"mfp2idx" gist (mfp2)

Table "herg_data.pairbvfps"

Column	Type	Modifiers
id	integer	not null
pairfp	bfp	
torsionfp	bfp	

Indexes:

"pairbvfps_pkey" PRIMARY KEY, btree (id)
"apfpbvidx" gist (pairfp)
"torsionfpbvidx" gist (torsionfp)

Matched pairs in SQL(simple)

```
select *,(pact1-pact2)/dist disparity,pact2-pact1 dact from
( select ms1.id id1,ms1.m smiles1,ms2.id id2,ms2.m smiles2,dist,
        -1*log(vs1.ic50_nm*1e-9) pact1,
        -1*log(vs2.ic50_nm*1e-9) pact2
  from ( select fp1.id id1,fp2.id id2,
                1.0-dice_sml(fp1.torsionfp,fp2.torsionfp) dist
    from cdk2.countfps as fp1
          cross join cdk2.countfps as fp2
      where fp1.torsionfp#fp2.torsionfp and fp1.id!=fp2.id
    ) cliff_pairs
  join cdk2.mols ms1 on (id1=ms1.id)
  join cdk2.mols ms2 on (id2=ms2.id)
  join cdk2.molvals vs1 on (id1=vs1.id)
  join cdk2.molvals vs2 on (id2=vs2.id)
  where dist>0
) tmp
where pact1>=pact2 and (pact1-pact2)>.1
order by disparity desc
```

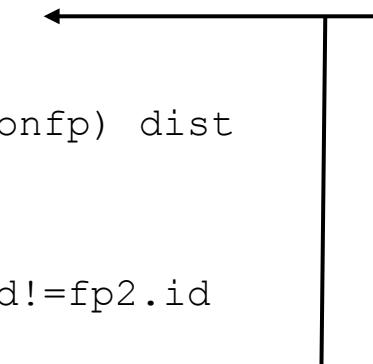
Matched pairs in SQL(simple)

```
select *,(pact1-pact2)/dist disparity,pact2-pact1 dact from
( select ms1.id id1,ms1.m smiles1,ms2.id id2,ms2.m smiles2,dist,
-1*log(vs1.ic50_nm*1e-9) pact1,
-1*log(vs2.ic50_nm*1e-9) pact2
from ( select fp1.id id1,fp2.id id2,
1.0-dice_sml(fp1.torsionfp,fp2.torsionfp) dist
from cdk2.countfps as fp1
cross join cdk2.countfps as fp2
where fp1.torsionfp#fp2.torsionfp and fp1.id!=fp2.id
) cliff_pairs
join cdk2.mols ms1 on (id1=ms1.id)
join cdk2.mols ms2 on (id2=ms2.id)
join cdk2.molvals vs1 on (id1=vs1.id)
join cdk2.molvals vs2 on (id2=vs2.id)
where dist>0
) tmp
where pact1>=pact2 and (pact1-pact2)>.1
order by disparity desc
```

← Generate pairs

Matched pairs in SQL(simple)

```
select *,(pact1-pact2)/dist disparity,pact2-pact1 dact from
( select ms1.id id1,ms1.m smiles1,ms2.id id2,ms2.m smiles2,dist,
         -1*log(vs1.ic50_nm*1e-9) pact1,
         -1*log(vs2.ic50_nm*1e-9) pact2
  from ( select fp1.id id1,fp2.id id2,
               1.0-dice_sml(fp1.torsionfp,fp2.torsionfp) dist
    from cdk2.countfps as fp1
         cross join cdk2.countfps as fp2
   where fp1.torsionfp#fp2.torsionfp and fp1.id!=fp2.id
  ) cliff_pairs
   join cdk2.mols ms1 on (id1=ms1.id)
   join cdk2.mols ms2 on (id2=ms2.id)
   join cdk2.molvals vs1 on (id1=vs1.id)
   join cdk2.molvals vs2 on (id2=vs2.id)
  where dist>0
) tmp
where pact1>=pact2 and (pact1-pact2)>.1
order by disparity desc
```



Calculate pIC50
Bring in smiles

Matched pairs in SQL(simple)

```
select *, (pact1-pact2) / dist disparity, pact2-pact1 dact from ← Calculate
( select ms1.id id1,ms1.m smiles1,ms2.id id2,ms2.m smiles2,dist, disparity
    -1*log(vs1.ic50_nm*1e-9) pact1,
    -1*log(vs2.ic50_nm*1e-9) pact2
  from ( select fp1.id id1,fp2.id id2,
                1.0-dice_sml(fp1.torsionfp,fp2.torsionfp) dist
    from cdk2.countfps as fp1
          cross join cdk2.countfps as fp2
    where fp1.torsionfp#fp2.torsionfp and fp1.id!=fp2.id
  ) cliff_pairs
  join cdk2.mols ms1 on (id1=ms1.id)
  join cdk2.mols ms2 on (id2=ms2.id)
  join cdk2.molvals vs1 on (id1=vs1.id)
  join cdk2.molvals vs2 on (id2=vs2.id)
  where dist>0
) tmp
where pact1>=pact2 and (pact1-pact2)>.1
order by disparity desc
```

Matched pairs in SQL (complete)

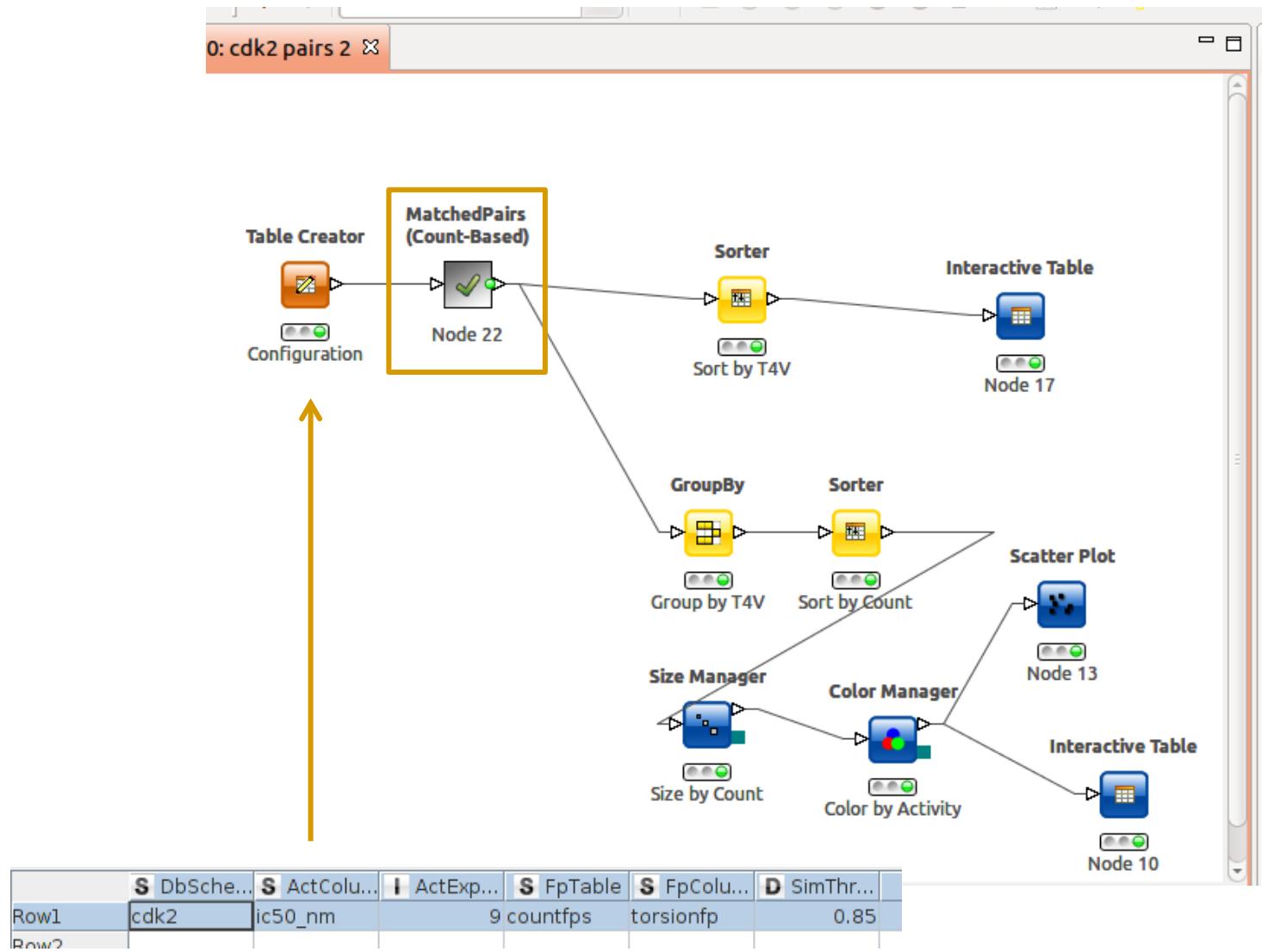
```
select *, (pact1-pact2)/dist disparity,pact2-pact1 dact from
( select ms1.id id1,ms1.m smiles1,ms2.id id2,ms2.m smiles2,dist,
        -1*log(vs1.ic50_nm*1e-9) pact1,
        -1*log(vs2.ic50_nm*1e-9) pact2,
        t4v_hash
  from ( select fp1.id id1,fp2.id id2,
               1.0-dice_sml(fp1.torsionfp,fp2.torsionfp) dist,
               md5(subtract(fp1.torsionfp,fp2.torsionfp)::text) t4v_hash
         from cdk2.countfps as fp1
         cross join cdk2.countfps as fp2
        where fp1.torsionfp#fp2.torsionfp and fp1.id!=fp2.id
      ) cliff_pairs
   join cdk2.mols ms1 on (id1=ms1.id)
   join cdk2.mols ms2 on (id2=ms2.id)
   join cdk2.molvals vs1 on (id1=vs1.id)
   join cdk2.molvals vs2 on (id2=vs2.id)
  where dist>0
) tmp
where pact1>=pact2 and (pact1-pact2)>.1
order by disparity desc
```

↑
**Label
transformations
based on
fingerprint math**

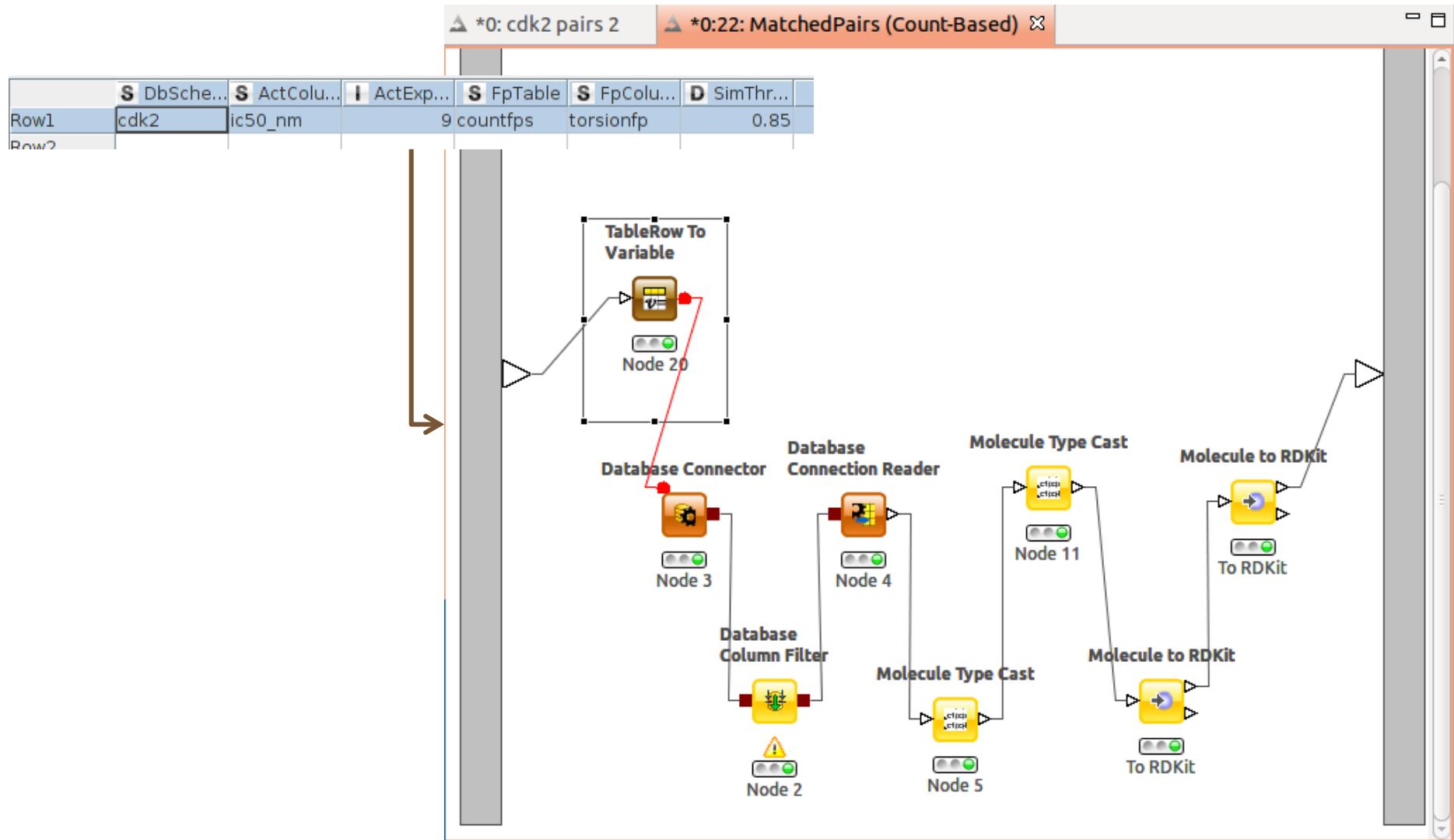
Performance

- Dataset: 1181 molecules with measured CDK2 IC50s
(source: binding db)
- Fingerprints: topological torsions (count-based)
- Counting results:
 - Similarity cutoff 0.90: 1400 pairs, 0.39 sec
 - Similarity cutoff 0.85: 3719 pairs, 0.53 sec
 - Similarity cutoff 0.75: 11541 pairs, 0.85 sec
- Retrieving results:
 - Similarity cutoff 0.90: 1400 pairs, 2.0 sec
 - Similarity cutoff 0.85: 3719 pairs, 4.9 sec
 - Similarity cutoff 0.75: 11541 pairs, 14.1 sec
- Hardware: Dell Studio XPS (i7 870, 64bit)

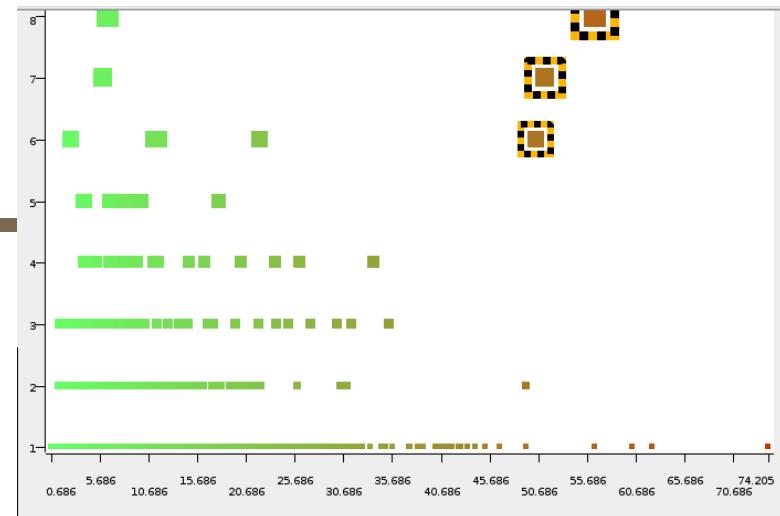
Knime implementation



Knime implementation

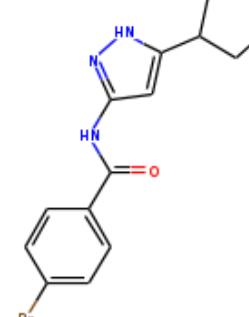
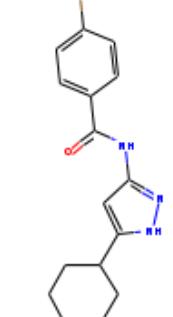
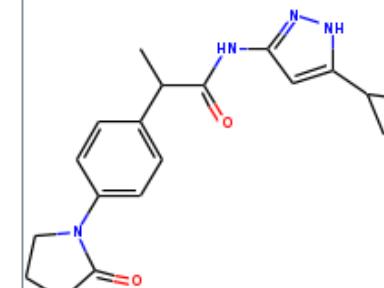
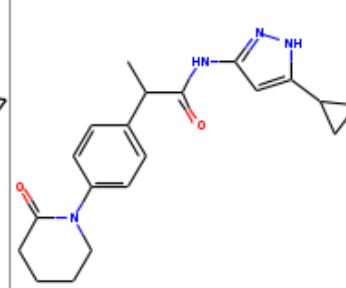
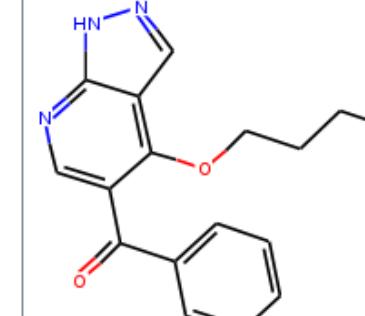
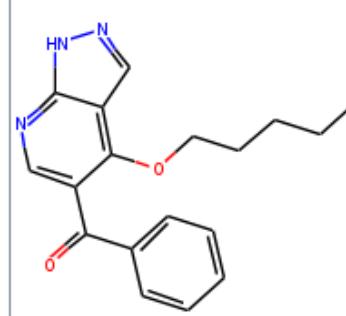


Knime implementation



Row ID	t4v hash	Mean(...)	First(mol1)	First(mol2)	Count(...)	Max(pact1)	Mean(dact)
Row2649	d3049eaf3c90e2ea202380a780a5b...	50.461			6	7.699	-0.463
Row937	4890255b136b311a2078798761e0...	51.404			7	7.824	-0.491
Row1523	78e01712a4f11b2e7adf2b33b21d0...	56.469			8	8.155	-1.053

Knime implementation

s t4v hash	d disparity	d dact	mol1	mol2
4890255b136b311a2078798761e00...	100.179	-1.301		
4890255b136b311a2078798761e00...	85.261	-0.812		
4890255b136b311a2078798761e00...	30.155	-0.339		

Knime implementation

S t4v hash	D disparity	D dact	mol1	mol2
78e01712a4f11b2e7adf2b33b21d067b	93.59	-1.586		
78e01712a4f11b2e7adf2b33b21d067b	69.981	-1.296		
78e01712a4f11b2e7adf2b33b21d067b	65.55	-1.214		

Wrapping up

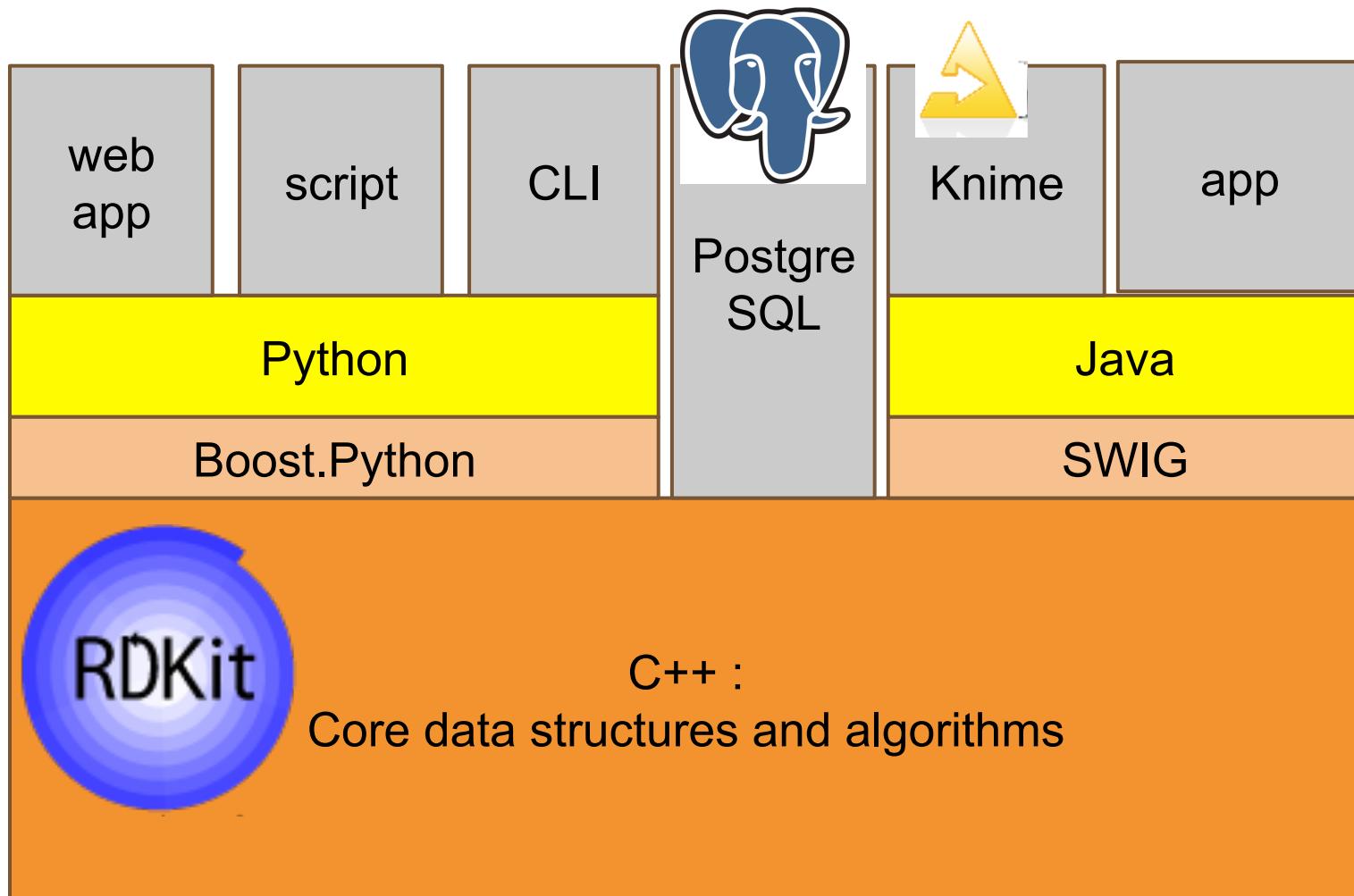
- What is it?
 - Cheminformatics toolkit useable from C++, Python, Java
 - Postgresql cartridge for substructure/similarity searching
 - Open-source Knime nodes for cheminformatics

- Web presence:
 - Main site: <http://www.rdkit.org>
 - Knime nodes: <http://tech.knime.org/community/rdkit>

We're hiring!

- Cambridge (the US one):
 - Cheminformatics developer in the Chemical Information Systems group
 - Technical lead in the Chemical Information Systems group
- If you're interested, please contact me

Thanks!



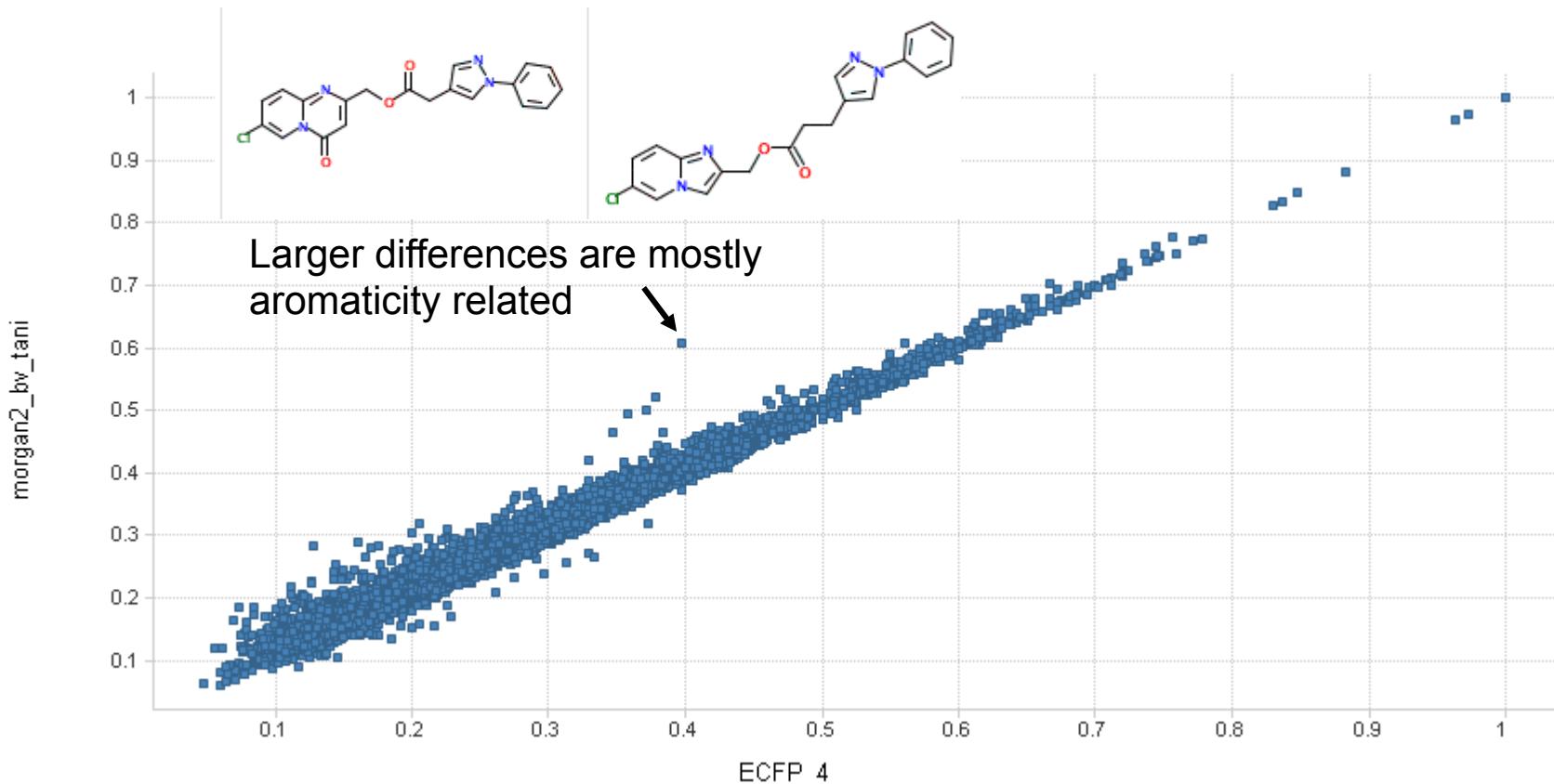
Backups

Comparing similarity measures

- Pick 10K random pairs of vendor compounds that have at least some topological similarity to each other (Avalon similarity ≥ 0.5)
- Compare similarities calculated with Pipeline Pilot and the RDKit

Comparing similarity measures

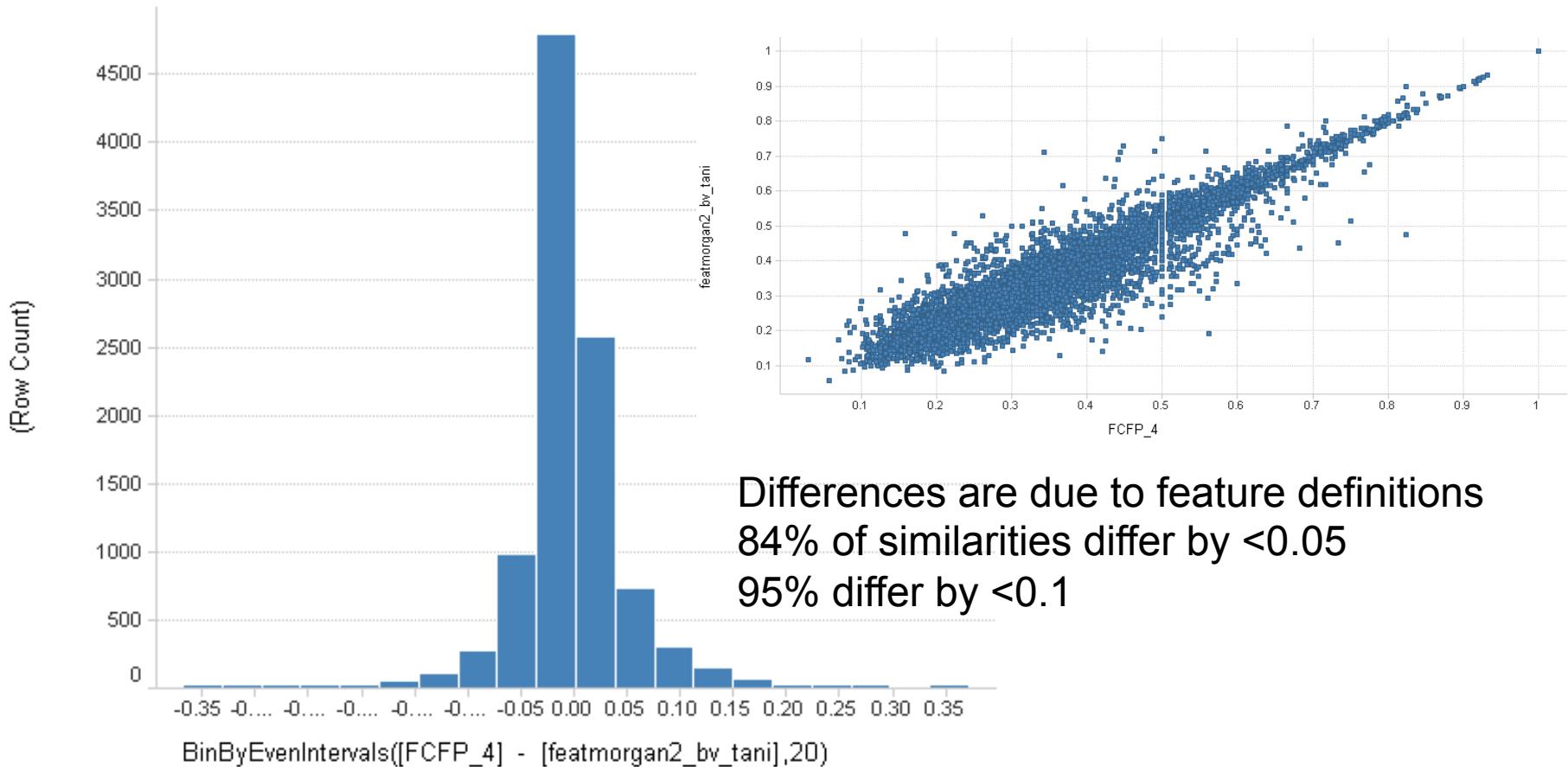
- RDKit Morgan2 vs PP ECFP4



- RDKit Morgan3 vs PP ECFP6 is similar

Comparing similarity measures

- RDKit FeatMorgan2 vs PP FCFP4



Contributing to open source: why bother?

- Scientific argument for releasing source:

ACS ethical guidelines: "A primary research report should contain sufficient detail and reference to public sources of information to permit the author's peers to repeat the work." (<http://pubs.acs.org/userimages/ContentEditor/1218054468605/ethics.pdf>)

Z. Merali, "ERROR: why scientific programming does not compute." *Nature* **467**:775-7 (2010)
N. Barnes, "Publish your computer code: it is good enough" *Nature* **467**:753 (2010)

Contributing to open source: why bother?

- Philosophy
- Improved code quality : users = testers/peer reviewers
- Gather new ideas/contributions from others
- Altruism : give something back to "the community"
- Selfishness : guarantee your own access to your work

Contributing to open source: why bother?

- We aren't the only big company doing this:
 - IBM
 - Apple
 - Google
 - Nokia
 - Microsoft(!)
 - many, many others
- We aren't even the only pharma company:
 - Sunesis
 - Lilly
 - Boehringer Ingelheim
 - Astra Zeneca
 - Sanofi Aventis
 - others

Practical Considerations

- Precompetitive
- Treat code publication process the same as publication of a scientific paper
- Pick a license carefully; use a standard one
- Management understanding and support
- Support from legal/patent department